TRABAJO FINAL DE CARRERA

TÍTULO DEL TFC: Sistema de monitorización y alarmas distribuido

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad Telemática

AUTOR: Javier Hernández Estévez

DIRECTOR: Toni Oller Arcas

FECHA: 10 de Marzo de 2009

Título: Sistema de monitorización y alarmas distribuido

Autor: Javier Hernández Estévez

Director: Toni Oller Arcas

Fecha: 10 de Marzo de 2009

Resumen

Este documento describe las tareas realizadas para la implementación de un sistema de control de alarmas distribuido basado en tecnologías estandarizadas que permiten asegurar la compatibilidad entre los dispositivos que lo integran. Se muestran pues el estudio y diseño de éste, así como la implementación y pruebas realizadas para el mismo.

Como resultado se obtiene una aplicación que, a través de la red, es capaz de descubrir sus semejantes, conformando una red lógica plenamente distribuida. En ésta los actores implicados integran un grupo de trabajo donde pueden interactuar entre sí, permitiendo hacer consultas o peticiones, y subscripciones a eventos.

El proceso de detección y descubrimiento se realiza mediante un sistema de mensajería, incorporando la notificación de alarmas o eventos. Por otro lado, tanto las peticiones como las subscripciones a alarmas se realizan mediante el uso de servicios Web.

Tanto el cliente como el servidor han sido implementados en C#, sobre el entorno de desarrollo Visual Studio .Net 2008. Posteriormente, para las pruebas realizadas se ha optado por el servidor IIS de Windows.

Title: Sistema de monitorización y alarmas distribuido

Author: Javier Hernández Estévez

Director: Toni Oller Arcas

Date: March, 10th 2009

Overview

This document describes the performed tasks to implement a distributed alarms control system based on standard technologies, ensuring compatibility among the integrated devices. It is showed then the study and design of this one, as well as the implementation and tests for it.

The result is an application which, via network is able to discover their peers forming a logical network fully distributed. In this one, related characters set up a work team that allows to interact with each other allowing to do requests or petitions and event subscriptions.

The discovery and detection process is done via a messaging system, adding alarms or events notification. On the other hand, both requests and alarms subscriptions are performed using web services.

Both client and server were implemented in C # on Visual Studio development environment. Net 2008. Afterwards testing was performed through Windows IIS server.

ÍNDICE

ÍNDICE FIGURAS

INTRODUCCIÓN

Hoy en día existen multitud de dispositivos y aplicaciones capaces de conectarse a una red y ser operables en ella. Las necesidades son muchas y variadas, y han impulsado el desarrollo de diferentes tecnologías y protocolos para su implementación, dejando así un escenario lleno de incompatibilidades debidas a las diferencias entre estos lenguajes o protocolos.

Esta situación ha creado la necesidad de investigar para el desarrollo de nuevos protocolos, que una vez estandarizados han salvado estas diferencias, surgiendo así una nueva generación de tecnologías para asegurar la portabilidad y compatibilidad entre sistemas. Ejemplos claros de estas tecnologías son las que se usan en el estudio de este proyecto.

El objetivo esencial de este proyecto no es más que el de familiarizarse con estas tecnologías y aprender a usarlas. Para ello se ha hecho un pequeño estudio de estas y se ha dispuesto un escenario en el que diseñar e implementar su uso. Este escenario supone una red local en la que se pueden hallar distintos dispositivos con capabilidad IP, como un sensor de temperatura o de movimiento, y otros dispositivos para la monitorización e interacción con estos, como una PDA o un portátil. Dichos dispositivos pueden estar diseñados con diferentes tecnologías o lenguajes, dada la naturaleza del proyecto.

Para el estudio de este proyecto se ha optado por el uso del entorno de desarrollo .NET y el lenguaje C#, desde donde se implementarán los dispositivos lógicos del proyecto, así como su funcionalidad e interacción.

En el primer capítulo encontraremos las especificaciones del proyecto, es decir objetivos, un ejemplo de uso y su análisis para posterior decisión del diseño.

En el segundo capítulo se encuentra ya el diseño del escenario del proyecto y sus dispositivos, así como el diseño de sus funcionalidades.

El tercero capítulo es una explicación de la implementación, en el entorno de desarrollo .NET, del diseño hecho en el tercer capítulo.

El capítulo cuatro corresponde a las pruebas realizadas para la comprobación y muestra del funcionamiento del proyecto implementado.

El capítulo cinco, contiene el cierre del proyecto: objetivos alcanzados, recuento de horas invertidas, aspectos a mejorar y conclusiones personales.

CAPÍTULO 1. ESPECIFICACIONES

En este capítulo se detallan los objetivos del proyecto, así como las consideraciones a tener en cuenta para el diseño del mismo.

1.1 Objetivos

El objetivo de este proyecto es el de implementar y estudiar un sistema de monitorización y alarmas distribuido.

Para ello será necesario conformar una red lógica de dispositivos que integren un grupo de trabajo, estos dispositivos pueden estar implementados con diferentes tecnologías y lenguajes por lo que será necesario trabajar sobre tecnologías estandarizadas que aseguren la compatibilidad entre dispositivos. Para conformar este grupo de trabajo, se diseñará un protocolo de descubrimiento.

Una vez ejecutado el proceso de descubrimiento y conformada la red lógica, dentro este grupo de trabajo los dispositivos participantes estarán en disposición de interactuar entre sí, pudiendo hacerse peticiones y subscripciones a los posibles eventos existentes.

Para la implementación de las peticiones y subscripciones de los dispositivos se hará uso de servicios Web, de esta manera se sigue objetivo principal del proyecto, asegurar la portabilidad y compatibilidad.

1.2 Ejemplo de uso

Se ha propuesto un ejemplo real como punto de partida para el estudio del proyecto, a partir del análisis de este ejemplo se decidirán los objetivos y puntos a seguir para el diseño e implementación de este.

En el ejemplo se cuenta con un conjunto de grandes instalaciones distribuidas, donde hay un gran número de dispositivos los cuales tienen diferentes funcionalidades y son capaces de negociar entre sí. También hay un conjunto más reducido de dispositivos destinados a la monitorización y gestión de los demás, estos corresponderían con los que poseen los operarios que trabajan en las instalaciones. A continuación se expone el ejemplo:

“Juan es un técnico del servicio de la red de saneamiento de Valencia que tiene como misión velar por el buen funcionamiento de las instalaciones de control y supervisión de las diferentes estaciones que gestionan la red del Ciclo Integral del Agua de Valencia. Su equipo de trabajo está formado por un mono de trabajo, un maletín con diversas herramientas y un **dispositivo** (por ejemplo una

PDA) que le permitirá interactuar remotamente con otros técnicos y con el propio sistema. El dispositivo tiene unas pequeñas capacidades multimedia (reproducción de contenido audiovisual, adaptado a las características del terminal) y permite disponer de **conectividad** mediante bluetooth, wifi o 3G, además de mecanismos de localización (GPS).

La compañía ha implantado servicios de **VoIP** en su plataforma. De esta forma, se podrán realizar **comunicaciones** (audiovisuales) utilizando la red propia de la compañía (IP), siempre que las condiciones de la red lo permitan. Este tipo de comunicaciones (llamadas telefónicas o videoconferencia) deben tener una **prioridad inferior** que las **comunicaciones de control** y **supervisión**.

En mitad de la jornada de trabajo Juan recibe una **llamada entrante** de su dispositivo. En el momento que Juan descuelga el teléfono, escucha una locución con el siguiente mensaje: “Se ha producido una incidencia, código ROJO, en sector A, en la estación remota de control de la Alameda “. Casualmente Juan estaba próximo a esta estación remota y el sistema ha determinado que Juan, por sus aptitudes (skills) y su localización era la persona indicada para resolver la incidencia. Mientras Juan se dirige al punto donde se ha producido el incidente la locución continúa: “Si quieres un itinerario que te dirija al destino pulsa “3”; Además el técnico puede recibir mensajes de texto en modo mensajería instantánea: “Hay una cámara que está registrando imágenes del incidente XXY en tiempo real, pulsa aquí para visualizar el incidente” o “Este incidente tiene un histórico de problemas asociados, pulsa aquí para acceder al historial del incidentes”. Al disponer de esta información directamente y antes de llegar al lugar exacto del problema puede filtrar o evaluar de manera avanzada la gravedad de la incidencia, activar un protocolo predefinido (por ejemplo llamada automática al 112). En ocasiones los propios operarios que están en la estación central, a pesar de tener una formación inferior también pueden resolver incidentes de manera remota, actuar sobre dispositivos remotos, etc.”

1.2.1 Análisis

En este ejemplo hay una red en la cual se dispone de diferentes dispositivos los cuales ofrecen diferentes servicios y pueden interactuar entre si. El desarrollo de este proyecto, se centrará en el estudio de un protocolo para la interacción entre estos dispositivos.

Este protocolo se basara en tecnologías que pretenden salvar problemas de incompatibilidad de sistemas, ya que de esta manera los dispositivos pueden estar implementados en múltiples plataformas y lenguajes.

Tal y como se aprecia en el ejemplo, existen los operarios, que corresponden con dispositivos que varían en la red. Este factor junto a la posibilidad de que los

dispositivos de la red también puedan variar hace necesaria la integración de un proceso de descubrimiento.

El descubrimiento es una parte esencial de este proyecto, ya que este se caracteriza por una red distribuida formada por dispositivos autónomos, de manera que es necesario algún sistema que ayude a definir y montar la red lógica de la cual formaran parte estos, y a partir de la cual podrán intercambiar información.

Por tanto, para que estos dispositivos formen una red distribuida, primero se tienen que conocer, de manera que se necesita un protocolo de descubrimiento para implementar esta función. Así, cuando estos dispositivos se conecten a la red, comenzará el proceso de descubrimiento, con el objetivo de dar a conocer su situación y funcionalidad, así como conocer la de los demás, conjuntando así una red plenamente distribuida y dinámica, en la cual los dispositivos son totalmente autónomos.

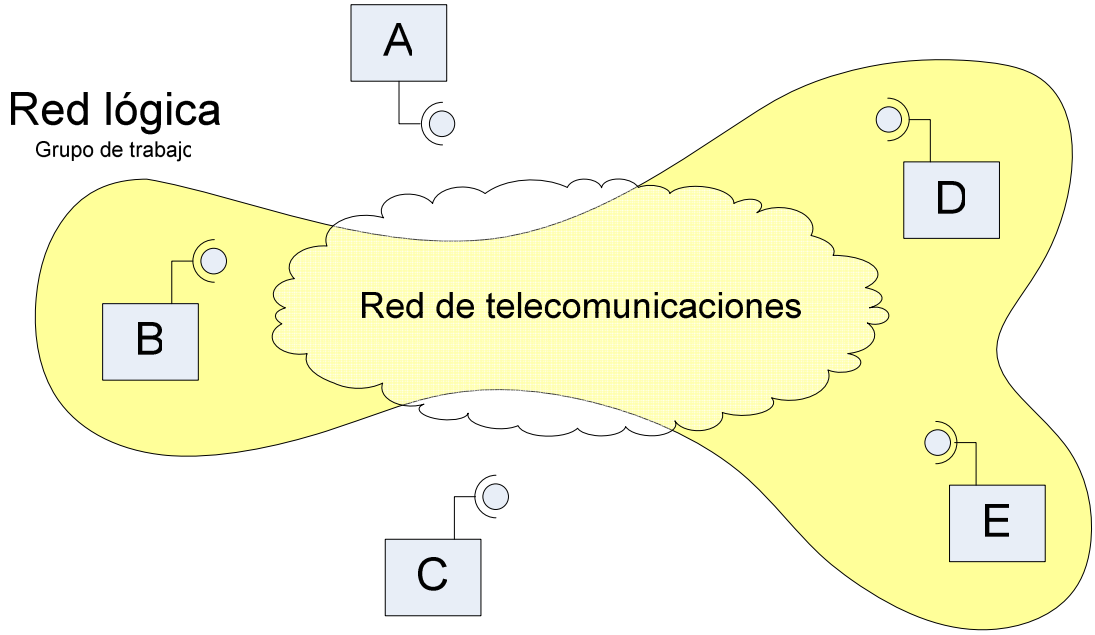

Fig. 1.1 Red lógica obtenida mediante el proceso de descubrimiento.

Una vez implementado el descubrimiento **(Fig. 1.1)**, cada dispositivo conoce la distribución de la red y sus funciones, de manera que ya están en disposición de comunicar-se, pudiendo hacer llamadas a los servicios que contiene cada uno, para ello se hará uso de tecnologías basadas en los servicios Web [3].

Estos servicios se dividirán en dos partes, por un lado las peticiones y por el otro las subscripciones, cuyas notificaciones se harán mediante la misma tecnología que el descubrimiento.

1.3 Tecnologías a tener en cuenta

Los servicios Web [1] deben ser independientes respecto a la opción de sistema operativo, modelo de objetos y lenguaje de programación para tener éxito en la disparidad de Web. Asimismo, para los servicios Web se aprovechen de la misma adopción extendida que otras tecnologías basadas en Web, deben ser: muy poco dependientes, contar con una comunicación ubicua y un formato de datos universal.

Además, los servicios Web emplean una infraestructura que proporciona lo siguiente: un mecanismo para localizar servicios Web, una descripción de servicio para definir cómo usar esos servicios y formatos de conexión estándar con los que comunicarse. La siguiente ilustración **(Fig. 1.2)** muestra un ejemplo de esta infraestructura.

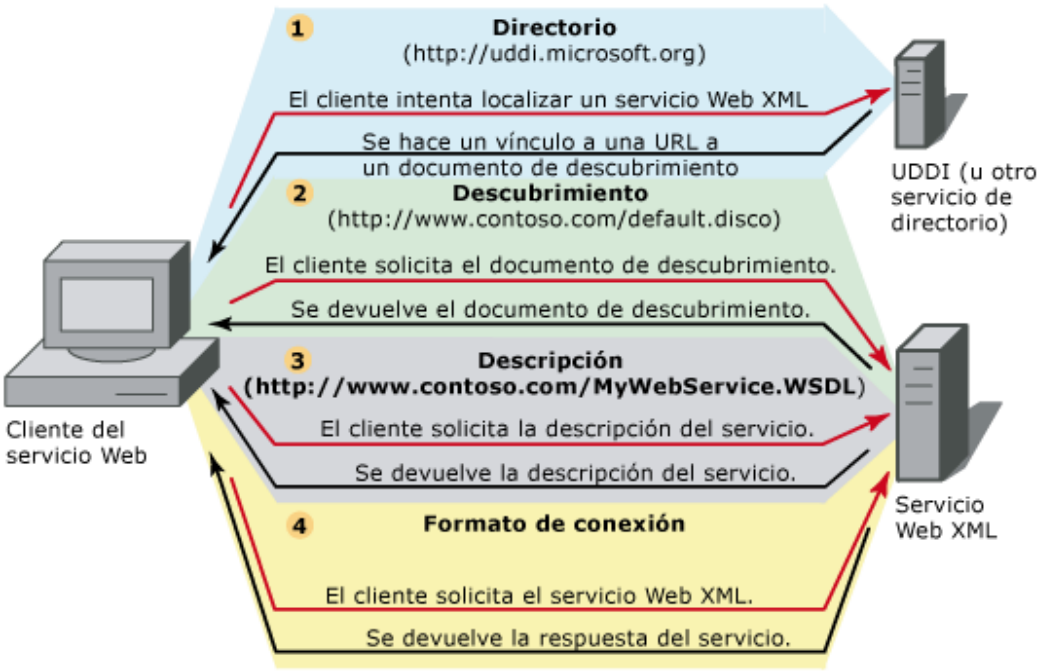

Fig. 1.2 Infraestructura servicios web

Para el desarrollo del proyecto, se aprovechará el proceso de descubrimiento de los dispositivos para integrar el de sus servicios Web.

Cada dispositivo actuará como directorio de sus propios servicios, de manera que compartiendo el url del archivo WSDL [5] en el proceso de descubrimiento de los dispositivos, los puntos 1 y 2 mostrados en la **figura 1.2** estarán solucionados. Así solo quedarán por solucionar los dos puntos restantes, 3 y 4 **(Fig. 1.2)**, cuya solución se haya en el desarrollo del **Capitulo 2.**

1.4 Dispositivos y funciones a implementar

Partiendo del análisis del ejemplo anterior **(Cap. 1.2)**, se han decidido implementar tres dispositivos diferentes para el estudio del proyecto. Por un lado se cuenta con una herramienta de gestión y monitorización de dispositivos, y por el otro dos dispositivos de tipo sensor.

1.4.1 Sensor de Movimiento

El primer dispositivo es un sensor Movimiento, este es el típico sensor para vigilancia y tiene un funcionamiento muy sencillo y básico, por lo que es el ideal para empezar. Su estado es de 1 o 0, según haya movimiento o no.

Para poder integrar este dispositivo en la red habrá que dotarlo de un protocolo de descubrimiento, así podrá darse a conocer y ser operable.

Una vez descubierto, los servicios que puede ofrecer este dispositivo son mínimos, tan solo se puede consultar el estado (movimiento o sin movimiento) o bien suscribir-se a este, en cuyo caso existen dos tipos de subscripciones, al movimiento o a la ausencia de este.

Una vez realizadas las subscripciones, cada cambio en el estado es comprobado, de manera que si ese estado contiene suscriptores, se les enviará una notificación de evento.

1.4.2 Sensor Temperatura

El siguiente dispositivo es un sensor Temperatura, este dispositivo corresponderá con un termostato o termómetro. Es algo más complejo que el de movimiento ya que cuenta con un valor numérico que indica el valor de la temperatura y tiene varios tipos de suscripción.

Como todos los demás deberá contar un sistema de descubrimiento para dar a conocer su funcionalidad, una vez publicados sus servicios estos podrán ser llamados remotamente por otros dispositivos.

Los servicios con los que contará serán de consulta y de suscripción, es decir, se podrá consultar la temperatura en tiempo real o bien suscribirse a un valor y condición en concreto.

Habrá tres tipos de suscripciones diferentes, la primera al valor de temperatura mayor que valor especificado, la segunda al valor de temperatura menor que valor especificado y la ultima al valor de temperatura igual que valor especificado.

Una vez realizada la suscripción, cada cambio de temperatura del sensor será comprobado para ver si coincide con alguna de las suscripciones, las que coincidan serán notificadas a sus pertenecientes mediante la misma mensajería que el descubrimiento, pero utilizando un mensaje de notificación de evento.

1.4.3 Monitor u Operario

Por último y como elemento más complejo, el Monitor u operario. Este será la herramienta de monitorización y gestión de dispositivos.

Como todos los otros dispositivos, contará con la herramienta de descubrimiento. Una vez realizado el proceso de descubrimiento, los dispositivos descubiertos se mostrarán en una interficie grafica, mostrando todas sus características principales: identificador, tipo, IP y url WSDL.

El siguiente paso una vez visualizados los dispositivos será el de seleccionar el deseado para visualizar los servicios que ofrece. Esto se llevará acabo mediante el uso del archivo WSDL [5] del servicio obtenido a través del URL compartido en el proceso de descubrimiento.

Una vez seleccionado el dispositivo a visualizar, se mostrarán los servicios que contiene, distinguiendo entre 3 tipos: uno para las consultas, otro para las subscripciones y otro para forzar los eventos de las subscripciones.

Para poder realizar una llamada a un servicio, se mostrarán los datos necesarios a introducir, una vez introducidos si así es necesario, se realizará la llamada al servicio y se mostrará el resultado de la respuesta.

Para los servicios de suscripción, cuando acontezca notificar un evento o alarma, esta se realizará mediante la misma tecnología que el descubrimiento, por lo que si se recibe una notificación se mostrará en pantalla.

CAPÍTULO 2. DISEÑO

Este capítulo contiene una descripción de los puntos a diseñar para el estudio del proyecto.

2.1 Mensajería

Para poder implementar el descubrimiento y todas las comunicaciones que no se realicen mediante los servicios Web, es decir Http y SOAP [4], se usara una mensajería propia. A esta mensajería se le añadirán los mensajes necesarios para la implementación del descubrimiento y de las notificaciones de evento. Esta librería se basa en XML [2], de esta manera mantenemos la premisa del proyecto de mantener una red totalmente portable y dinámica.

2.2 Descubrimiento

El descubrimiento es parte esencial del proyecto, ya que necesitamos formar de alguna manera una red lógica de dispositivos sobre los que trabajar.

Cuando un dispositivo se conecta a la red, envía un paquete de descubrimiento a una dirección multicast, a la cual pertenecen todos los dispositivos. La conexión se implementara sobre UDP, este protocolo es el mas apropiado par el caso, ya que nos facilita la comunicación que consta de mensajes pequeños que contienen la información esencial para el descubrimiento, a más TCP no implementa multicast.

Cuando el resto de dispositivos que forman la red reciben el mensaje de descubrimiento, del nuevo dispositivo, almacenan la información recibida, conociendo así la ubicación (dirección) i la funcionalidad de este (WSDL [5]). Y responden con otro mensaje que contiene la misma información, pero referente a ellos.

Así, el nuevo dispositivo se ha dado a conocer, “se ha descubierto” y conforme vaya recibiendo diferentes mensajes de los otros dispositivos, ira “descubriendo” quien forma la red.

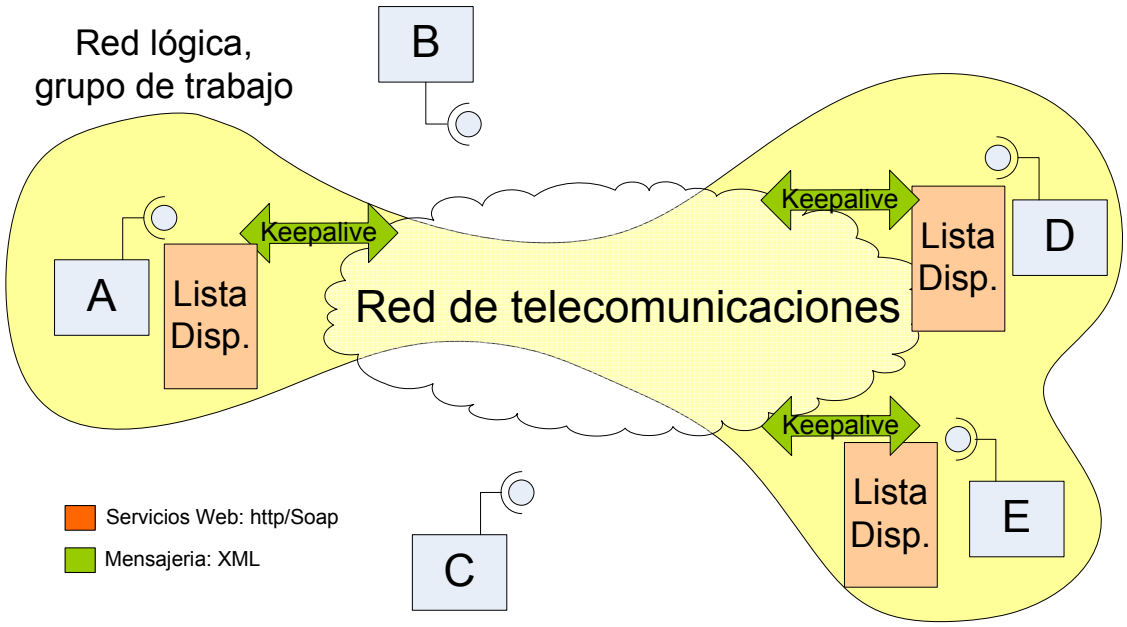

Fig. 2.1 Mantenimiento descubrimiento, red lógica.

Una vez finalizado este proceso, se pasara a hacer un "mantenimiento del descubrimiento" **(Fig. 2.1)**, es decir, se enviara un mensaje de keepalive con un periodo específico, 30seg, para indicar que el dispositivo sigue conectado a la red. De esta manera solucionamos la desconexión, así, si un dispositivo no recibe el keepalive de otro dispositivo conocido durante un tiempo limite, 90seg, eliminara este de la memoria, dando así por supuesto que ese dispositivo ya no forma parte de la red.

2.3 Servicios

Para hacer uso de servicios web, es necesario un proceso de descubrimiento del cual se obtiene un fichero WSDL [5] que contiene una descripción del servicio. Este proceso se implementara con nuestro protocolo de descubrimiento, publicando en la información de cada dispositivo el URL al archivo WSDL del servicio.

Partiendo de este punto hay que diferenciar dos partes en el proceso de consumo de un servicio, el cliente y el servidor, hablaremos de cliente para referirnos al operario que consume el servicio, y servidor para el dispositivo que lo ofrece.

Así pues, para la parte del cliente, será necesaria la generación de un Proxy [6] para el uso del servicio Web, este se puede generar a partir del URL obtenido del dispositivo y que hace referencia al archivo WSDL del servicio en cuestión.

2.3.1 Peticiones

Para las peticiones, una vez generado el Proxy, solo será cuestión de hacer una llamada a un procedimiento desde este, aportando los datos necesarios. El Proxy se comunicara con el servicio Web usando Http/SOAP [4], y este nos devolverá la respuesta.

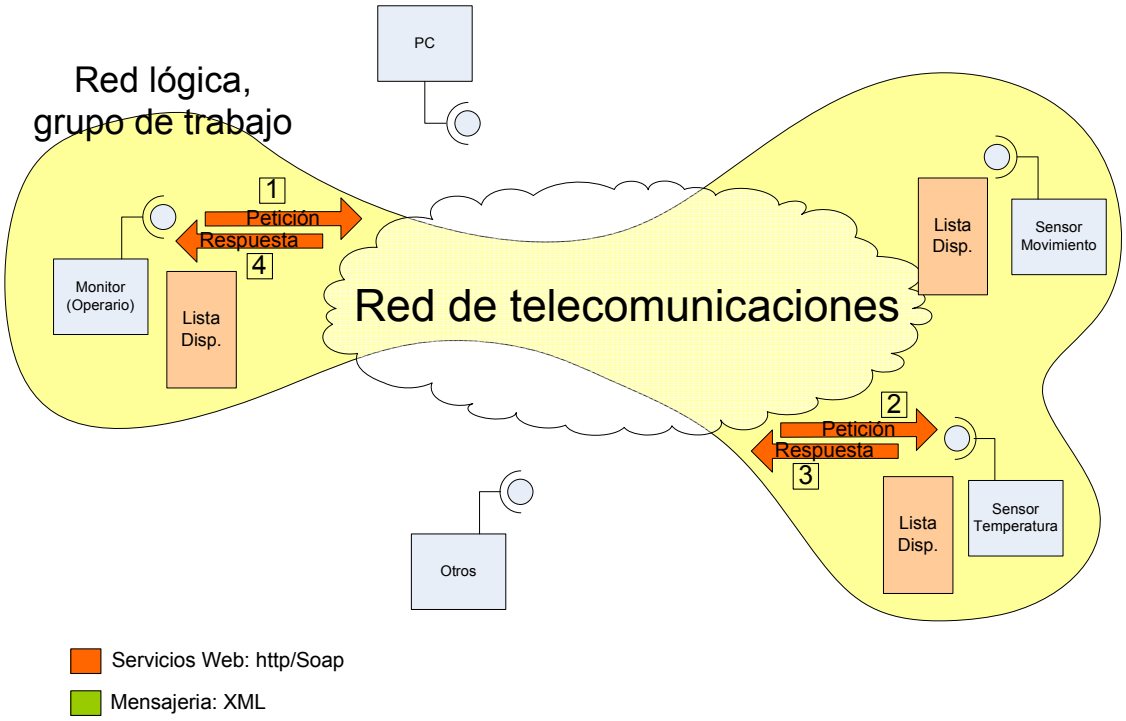

Fig. 2.2 Proceso de petición

2.3.2 Subscripciones

En las subscripciones el procedimiento inicial será el mismo que para las peticiones, primero se hará una llamada a un procedimiento desde el Proxy [6] al servicio Web, y este pasara la petición de suscripción a una pequeña aplicación que gestionará las subscripciones, puntos 2 a 5 (**Fig. 2.3)**.

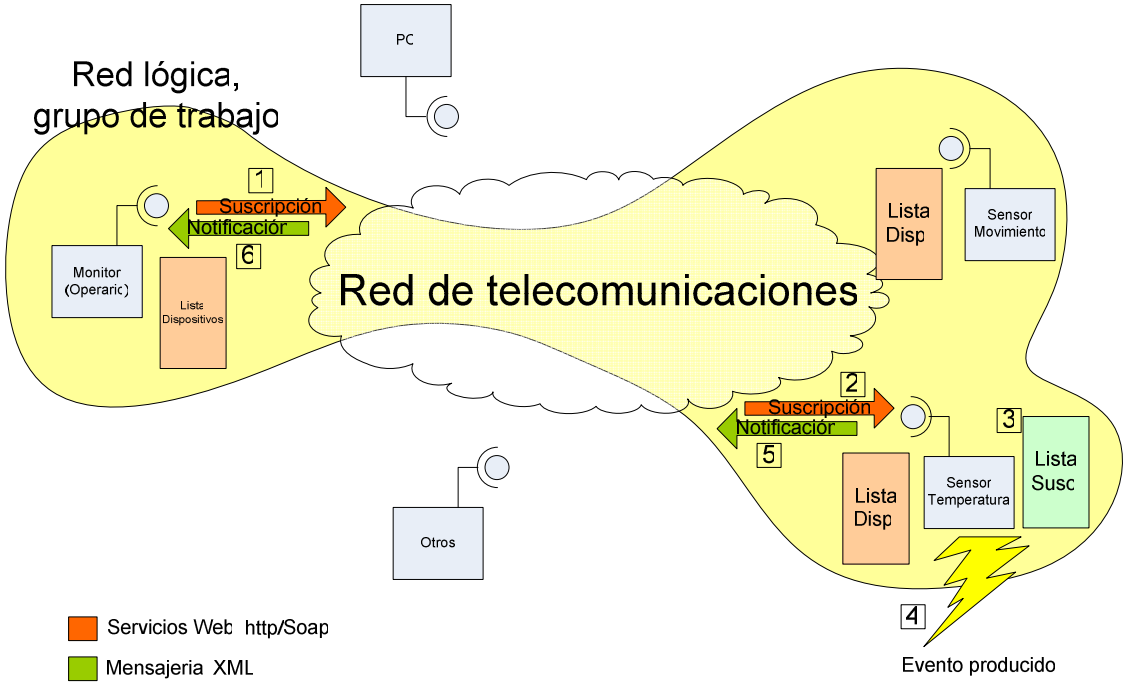

Fig. 2.3 Proceso Suscripción y notificación de evento

Esta aplicación almacena los datos de las subscripciones y comprueba que cuando haya un evento, si este corresponde con alguna suscripción, se le notifique mediante una notificación al dispositivo subscrito, puntos 4 a 6 **(Fig. 2.3)**. Para nuestro caso el operario.

2.3.3 Notificaciones

Las notificaciones se envían al dispositivo subscrito mediante la mensajería XML [2] **(Fig. 2.3)**, los mensajes enviados contienen información referente al evento sucedido.

CAPÍTULO 3. IMPLEMENTACIÓN

En este capítulo se describen las tareas realizadas para la implementación del diseño anterior. Para ello haremos uso del entorno .NET, éste nos ofrece muchas posibilidades a la hora de programar y además existe un amplio volumen de información al respecto.

3.1 Preparación

Para empezar nos hemos hecho con una mensajería en C# **(Fig. 3.1)**. Esta consta de dos proyectos, *Protocol* y ProtocolClasses. Estos dos proyectos se estructuran de la siguiente manera:

ProtocolClasses contiene 3 carpetas:

- Info, para alojar las clases que contendrán los datos a manejar.
- Messages donde se alojaran las clases que corresponderán con cada mensaje que creemos nuevo.
- Tipos serializables, para la serialización XML de clases *IP* y *Dictionary*.

Por otro lado, Protocol contiene también dos carpetas:

- *BusinessLogic*, donde se alojara la lógica del programa.
- Integration, donde tenemos un archivo llamado *NetworkManager.cs* que contiene los métodos necesarios para enviar y recibir los mensajes en nuestra aplicación. Aquí también se aloja una carpeta llamada *network* con la que trabaja *NetworkManager.cs* y que contiene todo el código referente a la red y necesario para la mensajería.

A partir de aquí se ha empezado a implementar nuestro protocolo. En la apartado *network* se ha substituido la serialización de la mensajería (*NetSerializationCoder.cs*) por una que trabaja con XML [2] (*XmlSerializationCoder.cs*), teniendo que añadir en *ProtocolClasses* dos clases mas para serialización de las clases *ipadress* y *dictionary*, que se hayan en *XML Serializable Types.*

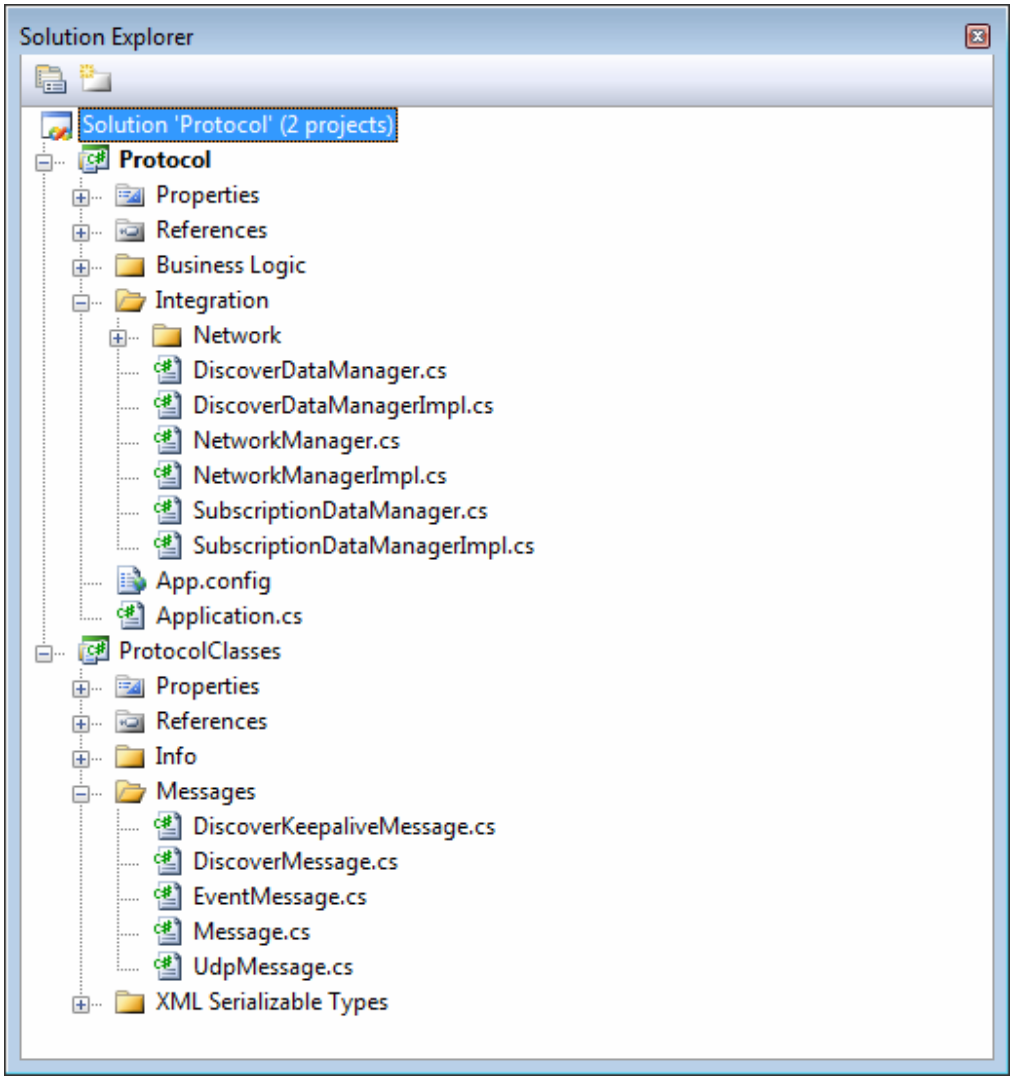

Fig. 3.1 Mensajería sobre la que vamos a trabajar.

3.2 Descubrimiento

Para la implementación del descubrimiento, en el proyecto *ProtocolClasses*, hemos añadido dos mensajes a la mensajería, *DiscoverMessage.cs* y *DiscoverKeepaliveMessage.cs*, en el apartado *Messages*. También, en el apartado *Info* se ha añadido *Constants.cs* para la declaración de constantes y se ha modificado el *DeviceInfo.cs* que sirve para manejar los datos de los dispositivos. Por otro lado en el proyecto *Protocol*, se han añadido las modificaciones necesarias al *NetworkManager.cs* para tratar los mensajes recibidos.

Toda la funcionalidad el descubrimiento se implementará en el proyecto *Protocol*. Se ha decidido dividir el tratamiento en dos capas, la primera solo hace de paso para los datos de la red (*NetworkManager.cs*) al protocolo de descubrimiento (*DiscoverManager.cs*), esta es *DiscoverDataManager* y se encuentra en el apartado *integration* junto a *NetworkManager*. La segunda se ubica en *BusinessLogic* y esta dividida en dos partes, una para gestionar la funcionalidad del protocolo dentro del propio dispositivo (*DiscoverManager.cs*) y la otra para gestionar los diferentes dispositivos con los que negocia a este (*DeviceManager.cs*).

Así, *DiscoverManager.cs* es quien lleva el peso del protocolo, recibe los datos del mensaje recibido a través de *DiscoverDataManager.cs* y decide como proceder. Los datos de los dispositivos que se van descubriendo son almacenados en *DeviceManager.cs*.

Para la implementación de la segunda capa, se han tenido que utilizar temporizadores, que han sido implementados con *threads*, uno para el envío de paquetes de *keepalive* dentro del *DiscoverManager.cs* y el otro para la desconexión de dispositivos en *DeviceManager.cs*.

Por otra parte el descubrimiento se lleva a cabo utilizando multicast, para evitar inundar la red de paquetes como seria el caso de la otra alternativa, broadcast. Por tanto, teniendo en cuenta que los dispositivos se encuentran todos dentro de una red local i todos tendrán que formar parte de un mismo grupo, solo utilizaremos una dirección multicast, a la cual pertenecerán todos los dispositivos (224.0.0.10).

Para poder implementar este tipo de comunicación, se han tenido que modificar los sockets [9] utilizados para UDP (*UdpTransport.cs*), dentro del apartado de red (*network*) de la mensajería.

Por último en el proyecto *Protocol*, se han aplicado los cambios necesarios a *Application.cs* para poner en marcha la aplicación. Llegados a este punto, contamos ya con una aplicación que cuenta con un protocolo de descubrimiento.

3.3 Interacción

Una vez realizadas las pruebas, pasaremos a distinguir entre dos implementaciones distintas, la del cliente u operario para el caso, y la del servidor o dispositivo.

Para la realización del dispositivo, se ha creado un servicio Web [6] y se le ha añadido el fichero *Global.asax* donde se puede manejar los eventos del servidor, como el comienzo y final de la aplicación o el inicio y final sesión. Se ha cargado la aplicación de descubrimiento en el evento Inicio de la aplicación

(*Start_Aplication*), de esta manera al iniciarse el servidor se activara el descubrimiento. Esto es debido a la naturaleza de los servicios Web, que consisten en activarse al recibir una petición, manejarla, responder si acontece y desactivarse. Así tenemos el descubrimiento en marcha, aunque el servicio este en reposo.

3.3.1 Dispositivo (Servidor)

Para la implementación del dispositivo se usaran dos tipos de dispositivos, un sensor de temperatura y otro de movimiento, ambos tienen un valor que podemos consultar y al que nos podemos suscribir. Para implementar este valor, se ha optado por crear una variable de tipo static int en el fichero *Global.asax*, esta se puede consultar y modificar.

Se han creado los siguientes WebMethods para publicar:

- Consulta de estado: devuelve valor entero, temperatura para el termostato y para el de movimiento 0 si no hay movimiento 1 si lo hay.
- Modificación de estado: se le pasa un entero, devuelve un string con el resultado.
- Suscripción:
    - Para el sensor de movimiento: se le pasa un entero que identifica al subscriptor y otro que identifica el estado al que subscribirse (0 o 1).
    - Para el sensor de temperatura: 3 casos, suscripción a >X <X o =X. se le pasa un entero que identifica al subscriptor y otro que identifica el valor.

Los Métodos Web de consulta y modificación, actúan sobre la variable static int que hay en *Global.asax*, pero para los de suscripción como la sesión puede haber expirado al producirse el evento, estos delegaran en nuestro protocolo inicial, donde implementaremos el trato de las subscripciones.

Para el trato de las subscripciones, se ha optado por montar un sistema parecido al del descubrimiento, es decir, la misma estructura. Para tratar los datos de las subscripciones contaremos con *SubscriptionInfo.cs* en el apartado *Info*, para almacenarlos con *DeviceSManager.cs* y para manejarlos *SubscriptionManager.cs*, ambos en el apartado de *BusinessLogic*.

Para recibir una petición de suscripción, esta se llevara a cabo mediante el servicio Web y este pasara la información a *SubscriptionManager.cs*, quien manejara la suscripción y la almacenara en *DeviceSManager.cs.* De esta manera, las subscripciones se irán almacenando para poder ser servidas cuando se produzca un evento.

Para poder producir un evento hemos creado un WebMethod en el servicio. Este llama a *SubscriptionManager.cs* para comprobar en *DeviceSManager.cs* si hay algún subscriptor de ese evento, si así lo fuera, envía un mensaje de evento al dispositivo subscritor.

De esta manera, se ha tenido que añadir también un Mensaje para los eventos llamado *EventMessage.cs* y añadir una capa (*SubscriptionDataManager.cs*) para el paso de datos entre *SubscriptionManager.cs* y *NetworkManager.cs* que es donde enviamos y recibimos mensajes.

3.3.2 Operario (Cliente)

Para la parte del cliente, partiremos del proyecto Windows Forms Aplication que usamos para las pruebas del descubrimiento. En este ya contábamos con una aplicación que efectuaba el descubrimiento, y nos mostraba los resultados. Se ha actualizado con los cambios realizados para el servidor, de manera que pueda procesar los mensajes de evento.

El siguiente paso es que nos muestre las peticiones y subscripciones que nos ofrece cada dispositivo, para ello haremos uso de los servicios Web.

Para programar con servicios Web en .NET, tenemos muchas facilidades, el inconveniente con el que nos encontramos es que para usar un servicio Web [6], hay que preensamblarlo, es decir el programador tiene que añadir una referencia al servicio Web antes de compilar el código.

Esto supone un problema, ya que contamos con diferentes servicios Web que no conocemos hasta estar en ejecución, es decir hasta haber realizado el descubrimiento y haber obtenido el URL al archivo WSDL [5]. Para ello se ha facilitado un link a un bloc, donde se explicaba como hacer una invocación dinámica de un servicio Web [7], de ahí obtuvimos una aplicación [8] a la cual, si le pasas un URL a un archivo WSDL, te genera la clase Proxy correspondiente. De esta aplicación se aprovecharán las dos librerías que llevan la funcionalidad de esta y las funciones necesarias para mostrar en la interficie grafica los datos que hay que introducir para cada método de la clase Proxy.

Una vez ensambladas las librerías en nuestro proyecto, se ha dispuesto de tal manera que el resultado es una interficie grafica dividida en dos secciones, una para el descubrimiento y listado de dispositivos, y otra para la interacción con los servicios que ofrecen.

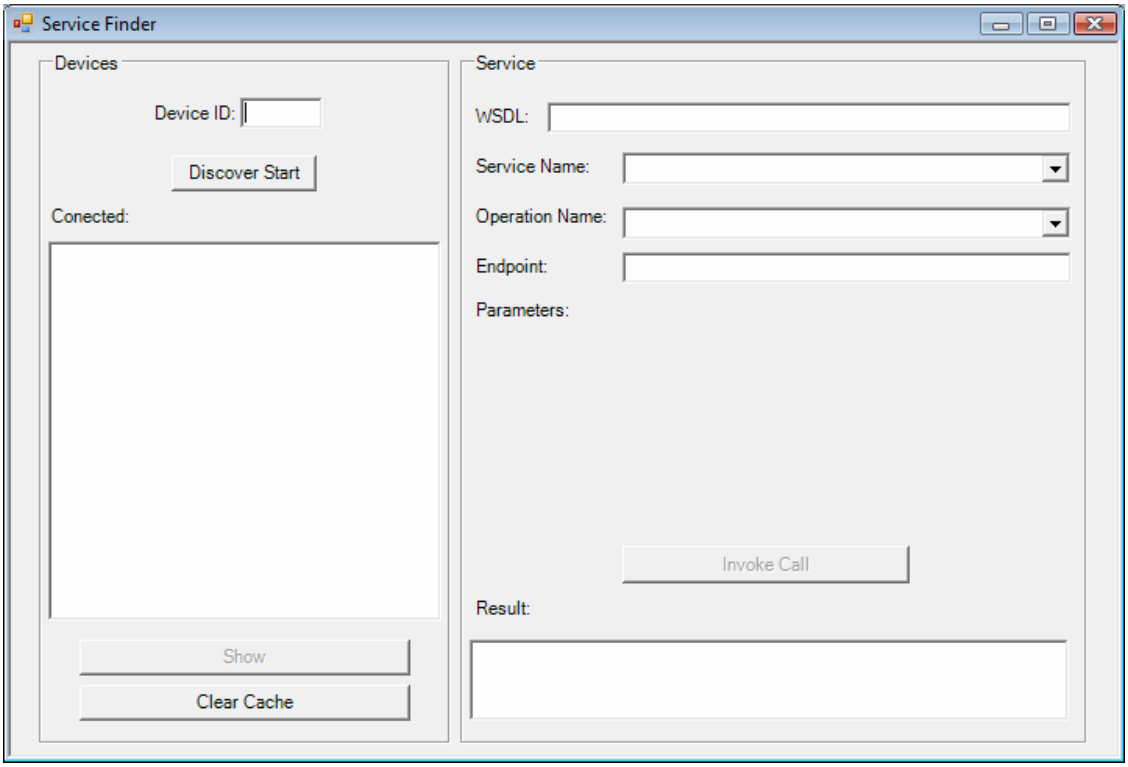

Fig. 3.2 Interficie grafica cliente u operario

La primera sección que corresponde con el grupBox Devices, contiene un botón “Discover Start”, el cual ejecuta la aplicación de descubrimiento. Luego tenemos un treeView en el que al clicar sobre el botón “Show”, situado debajo de este, se listan los dispositivos descubiertos y sus características. Una vez tenemos listados los dispositivos descubiertos, haciendo doble clic sobre el deseado, se mostrarán los servicios disponibles en la segunda sección.

La segunda sección, que correspondería con el grupBox Services, muestra las características de los servicios que ofrece el dispositivo seleccionado mediante doble clic en la primera sección, empezando por el URL del archivo WSDL [5] obtenido en el descubrimiento, y a partir del cual se han obtenido los servicios disponibles. Luego consta de dos comboBox, el primero muestra los servicios y según la selección de este, se muestran sus métodos publicados en el siguiente. Este otro, según el método seleccionado mostrara en un panel los parámetros necesarios para el uso del método, y un textBox para cada parámetro. Por ultimo tenemos un botón “Invoke Call” que invoca una llamada al método seleccionado, como si se tratara de un RPC normal, y muestra la respuesta en el cuadro “result”.

CAPÍTULO 4. PRUEBAS

Una vez realizada la implementación de los dispositivos, se han pasado a realizar las pruebas pertinentes para la comprobación y muestra del funcionamiento del protocolo y los servicios diseñados.

4.1 Protocolo descubrimiento

Para comprobar el funcionamiento del descubrimiento, se ha optado por una aplicación con interficie grafica que ejecuta el descubrimiento y lista los dispositivos descubiertos **(Fig. 4.1)**. También contamos con un log **(Fig. 4.2)** que nos indica todos los pasos que se van realizando según le hayamos indicado, es en este log donde podemos comprobar el intercambio de mensajes y manejo de dispositivos que se realiza en el descubrimiento.

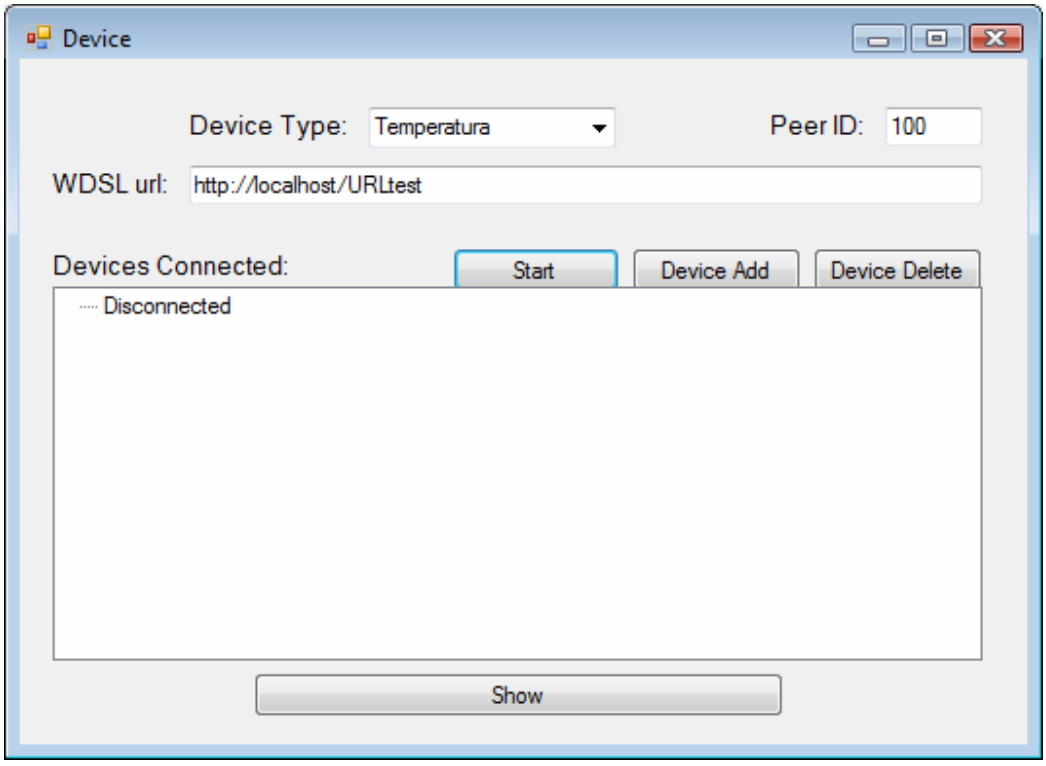

Fig. 4.1 Interficie gráfica para las pruebas del descubrimiento

Como podemos ver en la figura **Fig. 4.1** tenemos una interficie grafica en la cual contamos con un botón de “Start” desde el que se lanza la aplicación usando los parámetros del dispositivo que introducimos arriba, estos corresponden al tipo e identificador de dispositivo, y al URL del WSDL del servicio.

Al clicar en “Start” empezará el proceso de descubrimiento y podremos comprobar como se van enviando los mensajes a través de log **(Fig. 4.2)**.

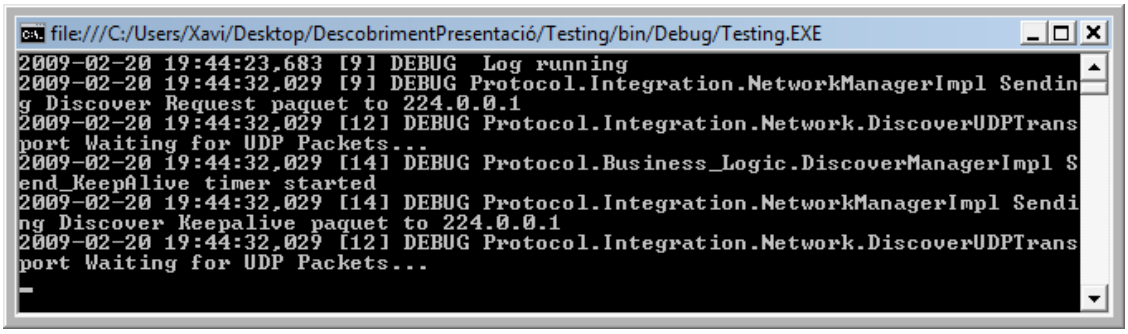

Fig. 4.2 Log para la comprobación del funcionamiento del código

Aquí podemos ver el log de la aplicación **(Fig. 4.2),** justo después de iniciarla mediante el botón start.

Una vez puesto en marcha este primer dispositivo, podremos poner otros dispositivos en marcha para comprobar su funcionamiento. Veremos que al descubrirse los nuevos dispositivos se notificaran con un messageBox, después, solo tendremos que clicar en el botón “Show” y se listaran en el treeview los dispositivos agregados y sus campos **(Fig. 4.3).**

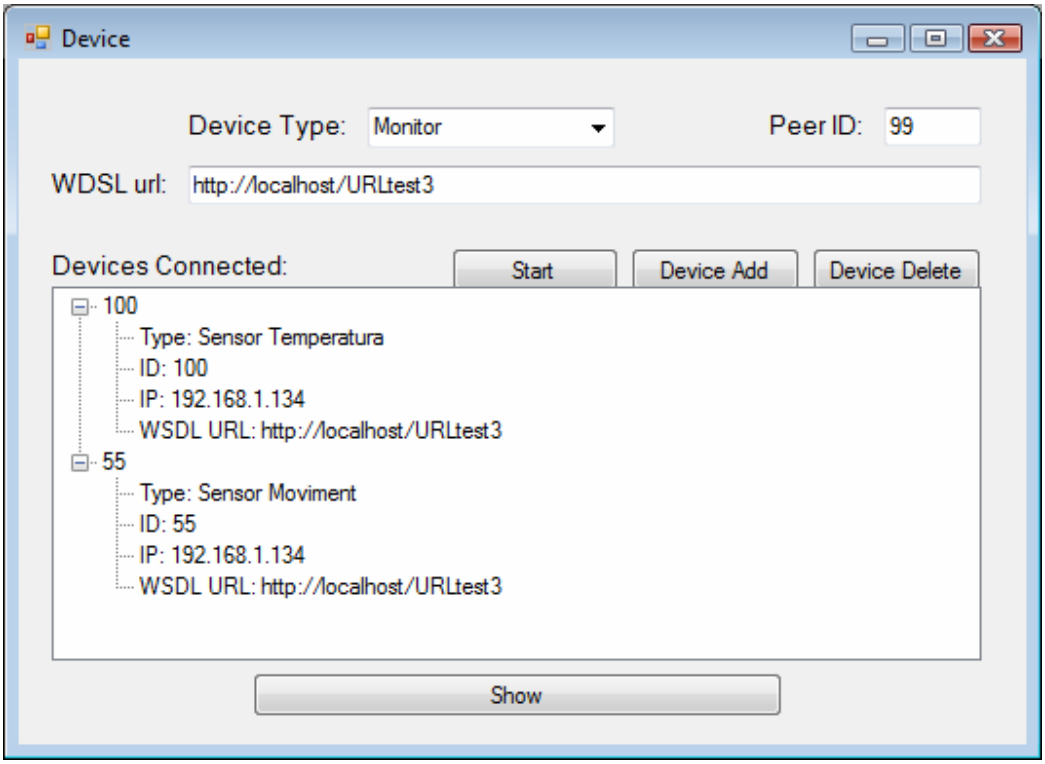

Fig. 4.3 Listado de dispositivos descubiertos.

4.2 Dispositivo (Servidor)

Para poder realizar las pruebas necesarias e implementar los dispositivos, se ha tenido que utilizar el IIS de Windows (Internet Information Server), ya que con ASP.NET el servidor entra en un modo “sleep” que trunca la continuidad de la aplicación por lo cual no es posible tener el descubrimiento en marcha.

Para el uso del IIS se ha tenido que instalar este desde programas de Windows. Una vez instalado, se ha publicado el servicio Web en el directorio root del servidor (IIS), a continuación, desde el administrador del IIS, en Application Pools se ha añadido uno de tipo ASP .NET, y en Web Sites se le ha agregado el servicio Web como aplicación usando el application pool añadido.

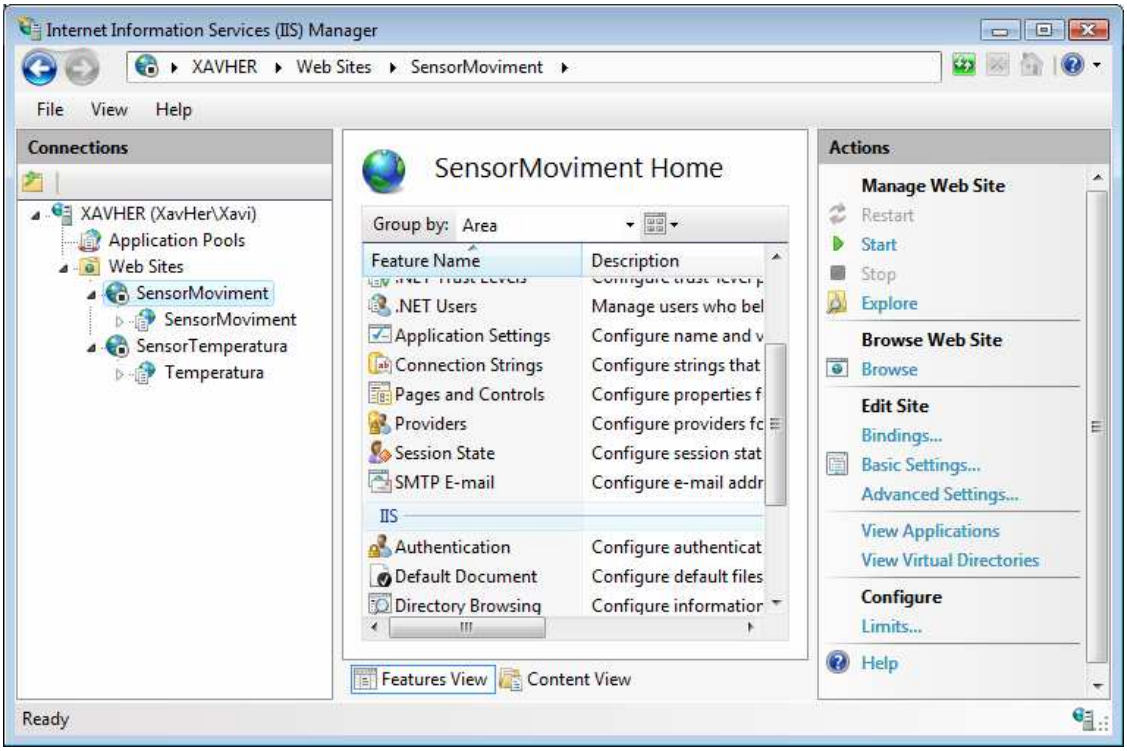

Fig. 4.4 Internet Information Server (IIS)

Después de poner el servidor en marcha se ha accedido a nuestra aplicación mediante un explorador, de esta manera se pretende forzar el comienzo de la aplicación.

Como se muestra en la imagen, se han generado dos dispostivos para su uso en las pruebas, un sensor de movimiento y uno de temperatura o termostato.

4.3 Monitor (Cliente)

Con la aplicación en marcha ya tenemos el descubrimiento funcionando y los servicios de ese dispositivo disponibles, por lo que pasaremos a trabajar solo con el cliente, para ello ejecutaremos la aplicación con interficie grafica del cliente y poniendo el identificador de dispositivo (del cliente) clicaremos sobre “Discover Start” para forzar el descubrimiento.

En este punto las capas de de descubrimiento de ambos dispositivos negociaran el descubrimiento, una vez realizado el proceso ambos dispositivos se conocerán y se lanzara un mensaje indicando que dispositivos se han agregado.

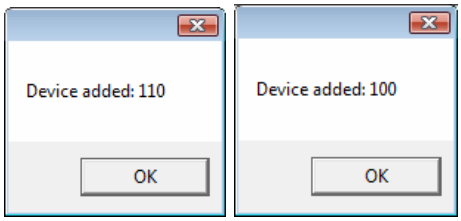

Fig. 4.5 MessageBox para notificación de nuevos dispositivos agregados

Después de forzar el descubrimiento, cuando se nos muestre un mensaje de dispositivo añadido, listaremos el contenido del descubrimiento, esto se consigue clicando sobre el botón mostrar de la sección de dispositivos del cliente **(Fig. 4.6)**.

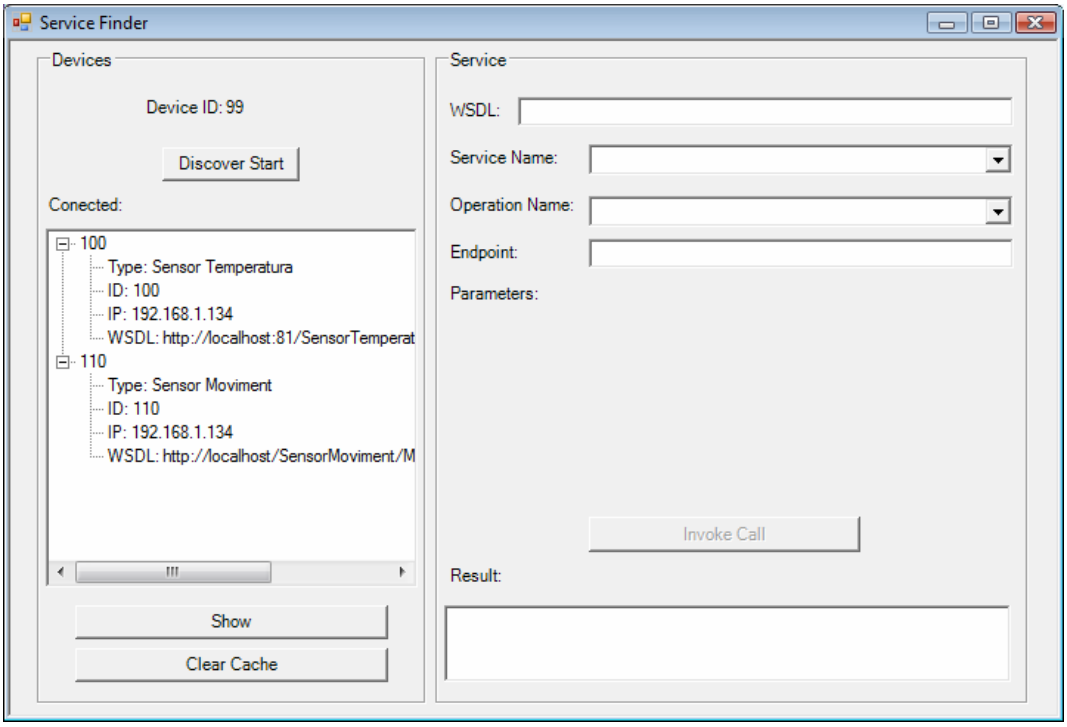

Fig. 4.6 Interfície grafica cliente con dispositivos descubiertos

Los dispositivos descubiertos se mostraran en un árbol de vistas donde haciendo doble clic sobre el dispositivo se pasarán a mostrar sus propiedades dentro de la sección de servicios **(Fig. 4.7).**

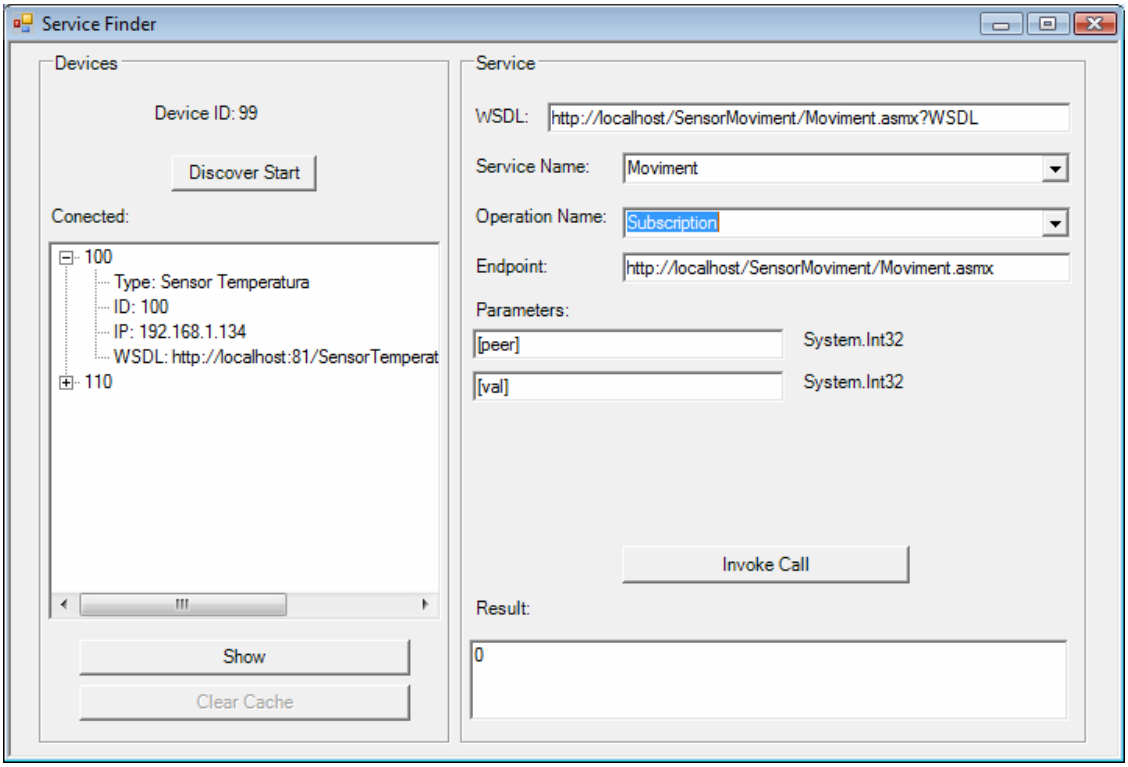

Fig. 4.7 Cliente con Proxy “Sensor Movimiento” en funcionamiento

Una vez generada la clase Proxy desde la que trabajaremos sobre el servicio, vemos que se nos muestran los servicios disponibles y sus métodos expuestos, “Operation Name” **(Fig. 4.7)**. Más abajo en la sección de parámetros, se mostrarán los parámetros necesarios para la llamada al método seleccionado. Por ultimo, si se clica en “Invoke Call” se efectuara la llamada al método, la clase Proxy negociara con el servidor usando http/SOAP [4] y obtendrá el resultado de la llamada mostrándolo en la sección result.

Para comprobar el funcionamiento de las subscripciones, nos subscribiremos a un evento en un dispositivo y forzaremos el suceso de este evento. Para ello escogeremos uno de los dispositivos, por ejemplo el de movimiento, y seleccionaremos el método subscription. Se nos mostraran dos parámetros a introducir **(Fig. 4.7)**, el identificador de dispositivo subscriptor y el valor al que deseamos suscribirnos, escribiremos 99 y 1 respectivamente, y nos dará como resultado Suscripción correcta.

El sensor de movimiento consta de un valor que puede variar entre 0 (sin movimiento) y 1 (movimiento), y se inicializa con 0. Por tanto nos hemos subscrito al movimiento. Para forzar el suceso de movimiento bastara con seleccionar el método producir evento e introducir el valor 1, esto forzara el evento en el dispositivo y este comprobara si tiene algún dispositivo subscrito a

ese evento, si es así nos enviará un mensaje de notificación de evento mediante la mensajería en XML [2] y nuestra aplicación mostrará un mensaje de evento producido en dispositivo x.

Para el funcionamiento del sensor de temperatura la diferencia es que este tiene un valor entero que variara según la temperatura y tiene 3 subscripciones distintas (valor ‘<’, ‘>’ y ‘=’ que T). El procedimiento para su prueba de funcionamiento es el mismo que el anterior.

CAPÍTULO 5. CONCLUSIONES

Realizadas las pruebas y comprobado el funcionamiento de los dispositivos implementados, se pasara a hacer una síntesis de los objetivos y conclusiones alcanzados, así como del tiempo invertido y las implicaciones medioambientales.

5.1 Objetivos alcanzados

Finalizada la realización del proyecto, el objetivo principal ha sido alcanzado: se ha estudiado y logrado implementar un sistema de monitorización y alarmas distribuido haciendo uso de las tecnologías XML [2] y Servicios Web [1].

Para ello primero se ha tenido que conformar una red lógica y distribuida, gracias al diseño de un protocolo de descubrimiento cuyo funcionamiento es muy sencillo y está basado en XML para asegurar otro de los principales objetivos del proyecto, el de la portabilidad y compatibilidad entre sistemas. Objetivo también asegurado por el uso de servicios Web en la implementación de las peticiones y de las subscripciones.

Así pues se han logrado los principales objetivos del proyecto, adquiriendo nuevos conocimientos de XML y sobretodo de los servicios Web, y aportando nuevas experiencias que mejorarán el desarrollo de futuros proyectos.

5.2 Horas invertidas

En este apartado intentaremos reflejar en un pequeño guión las horas invertidas para el desarrollo del proyecto.

TEMA	COMENTARIOS	HORAS
Estudio previo y especificaciones	Conceptos, objetivos y conocimientos a adquirir	20h
	Lectura de otros TFCs como referencia	10h
	Servicios Web: Estudio	20h
	IIS: Estudio para ejecución de aplicación.	10h
	Mensajería: Estudio código	20h
Diseño	Protocolo Descubrimiento	6h
	Peticiones	10h
	Subscripciones	20h
Implementación	Mensajería: Adaptación según necesidades	30h
	Protocolo Descubrimiento	40h
	Invocación dinámica de servicios Web [7] [8]. Estudio y adquisición de conocimientos, adaptación código adquirido.	50h
	Protocolo descubrimiento: Integrarlo en la aplicación cliente y servidor	30h

	Peticiones: Implementación	20h
	Subscripciones: Implementación	50h
Pruebas	Testeo aplicación y dispositivos	10h
	Reajustes	20h
Conclusiones	Documentación: Redacción memoria	60h
Documentación	Documentación: Redacción y preparación presentación	20h
Proyecto	**Total horas**	**446h**

Tab. 5.1 Horas invertidas

El siguiente diagrama muestra el tiempo de dedicación durante el transcurso del proyecto. Las tareas descritas en la tabla **Tab. 5.1** se distribuyen de manera no consecutiva a lo largo de todo el periodo.

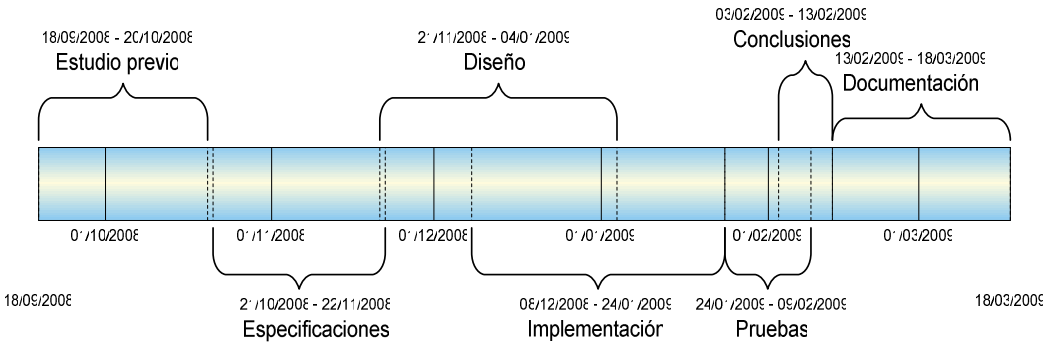

Fig. 5.1 Tiempo de dedicación

5.3 Posibles mejoras

Como posibles mejoras del proyecto haría falta destacar la posibilidad de implementar el descubrimiento y las notificaciones con Http/SOAP [4] para integrar todo el proyecto bajo una sola tecnología estandarizada. Otras posibles mejoras serían implementar más dispositivos incluso con otros sistemas o lenguajes de programación, ya que la naturaleza del proyecto así lo permite.

También se podrían enfocar las mejoras hacia el terreno multimedia, pudiendo incluir cámaras por ejemplo, como dispositivos adicionales para la seguridad. O bien enfocarlo hacia telefonía IP (VoIP), como en el ejemplo de uso, que también se tienen en cuenta prioridades en la gestión de los tráficos, otro punto también a mejorar.

5.4 Estudio del impacto medioambiental

Para el estudio del impacto medioambiental del proyecto, nos fijaremos en los tiempos y frecuencias de envió de los paquetes de descubrimiento, ya que es en este punto donde el diseño del proyecto puede variar el consumo de energía, aunque este sea mínimo, variando también el impacto medioambiental.

De esta manera se ha impuesto un tiempo de 30 segundos para reiteración en el envío de paquetes de descubrimiento, considerando este tiempo suficiente para mantener el descubrimiento en tiempo real y no hacer envíos innecesarios.

Por otro lado, hay que tener en cuenta que se trata de un proyecto donde la gestión y funcionalidad de los elementos no es centralizada, por lo que el consumo de energía se distribuye.

Otro factor a tener en cuenta, es que el proyecto se desarrolla sobre grandes superficies de instalaciones, donde la gestión y monitorización de los elementos se efectúa de manera remota. Esto implica que los empleados no tengan que desplazarse para solventar incidencias, reduciendo así el consumo de gasolina, lo que disminuye el impacto medioambiental.

Este factor también implica la ausencia de personal en zonas donde es necesaria la intervención constante de este, reduciendo así el consumo de bienes y servicios de los inmuebles, como puede ser el agua, la luz, calefacción o aire acondicionado. Todo este conjunto de ahorros en diferentes materias tiene como consecuencia la disminución del impacto medioambiental, provocado precisamente por el proceso de obtención o consumo de estas.

5.5 Conclusiones personales

La realización de este proyecto a nivel personal ha sido muy satisfactoria y gratificante, me ha aportado nuevas experiencias en el desarrollo de proyectos, sobre todo a nivel individual, ya que es el primer proyecto de esta envergadura que abordo yo solo.

He adquirido nuevos conocimientos sobre los servicios Web ampliamente utilizados, que me han despertado mucho el interés sobre este tema, aunque me han dado algunos quebraderos de cabeza, al encontrarme con algunas dificultades referentes al entorno de desarrollo Visual Studio .NET, pero esto me ha ayudado a ampliar mis conocimientos de este.

El hecho de tener que implementar y probar el funcionamiento del proyecto, y ver que tiene una aplicación real y palpable muy extensa, ha hecho que la experiencia sea más amena y enriquecedora, y que crezca el interés en el desarrollo del proyecto.

Otro factor a destacar es el de la planificación, recalcar que es importante tener una planificación mínima preparada, ya que ante imprevistos podemos resultar más efectivos. Imprevistos como los que me he podido encontrar yo: errores insalvables de Windows con la consecuente reinstalación de este, o cambios drásticos en el diseño e implementación del proyecto.

Por último resaltar que este proyecto pretende aportar los conocimientos necesarios y hacer una familiarización sobre las tecnologías XML y de los Servicios Web, permitiendo así la implementación de este con otras o varias tecnologías. Por esa misma razón se ha optado por un diseño sencillo de los dispositivos a implementar.

ACRONIMOS

PDA

Personal **D**igital **A**ssistant (Asistente Digital Personal), es un computador de mano originalmente diseñado como agenda electrónica.

3G

Es la abreviación de tercera-generación en telefonía móvil. Los servicios asociados con la tercera generación proporcionan la posibilidad de transferir tanto voz y datos (una llamada telefónica) y datos no-voz (Descargas, etc.)

GPS

Global **P**ositioning **S**ystem, es un Sistema Global de Navegación por Satélite que permite determinar en todo el mundo la posición de un objeto o dispositivo.

VoIP

Voz sobre **IP**, Es el encaminamiento de conversaciones de voz sobre Internet o sobre cualquier otra red basada en IP.

IP

Internet **P**rotocol: dirección IP, el número que identifica a cada dispositivo dentro de una red con protocolo IP. Protocolo IP, un protocolo usado para la comunicación de datos a través de una red.

XML

e**X**tensible **M**arkup **L**anguage. Lenguaje de marcado ampliable o extensible. Desarrollado por el World Wide Web Consortium (W3C).

SOAP

Simple **O**bject **A**ccess **P**rotocol: es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

WSDL

Web **S**ervices **D**escription **L**anguage, un formato XML que se utiliza para describir servicios Web.

URL

Uniform **R**esource **L**ocator, Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.

HTTP

HyperText **T**ransfer **P**rotocol es el protocolo usado en cada transacción de la Web

UDP

User **D**atagram **P**rotocol, es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión

TCP

Transmission **C**ontrol **P**rotocol, es un protocolo de transporte basado en segmetos, establece conexiones para luego transmitir flujos de datos.

IIS

Internet **I**nformation **S**erver, es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios. Actúa como servidor web.

RPC

Remote **P**rocedure **C**all, es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos

BIBLIOGRAFÍA

[1] **Servicios Web:** documentación de referencia. Informe de **Benjamín González C.** Ingeniero de Sistemas. [Última consulta: 15 Octubre 2008] URL: http://www.desarrolloweb.com/articulos/1535.php

[2] **XML:** documentación de referencia. *Wikipedia.org, la enciclopedia libre*. [Última consulta: 20 Octubre 2008] URL: http://es.wikipedia.org/wiki/XML

[3] **Servicios Web:** documentación de referencia. Copyright © 1994-2005 W3C® (MIT, ERCIM, Keio). [Última consulta: 20 Octubre 2008] URL: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

[4] **SOAP:** documentación de referencia. *Wikipedia.org, la enciclopedia libre*. [Última consulta: 20 Octubre 2008] URL: http://es.wikipedia.org/wiki/SOAP

[5] **WSDL:** documentación de referencia. *Wikipedia.org, la enciclopedia libre*. [Última consulta: 20 Octubre 2008] URL: http://es.wikipedia.org/wiki/WSDL

[6] **Servicios Web:** ejemplos y explicaciones de uso en todo lo que se refiere a implementación. [Última consulta: 15 Noviembre 2008] URL: http://msdn.microsoft.com

[7] **Invocación dinámica servicios Web:** ejemplos de uso. Thursday, April 27, 2006 8:00 AM by kaevans. [Última consulta: 18 Diciembre 2008] URL: http://blogs.msdn.com/kaevans/archive/2006/04/27/dynamically-invoking-a-web-service.aspx

[8] **Invocación dinámica servicios Web:** aplicación con código abierto. 2002-2008 by thinktecture, Ingo Rammer and Christian Weyer. Última consulta: 10 Enero 2008] URL: http://www.thinktecture.com/resourcearchive/tools-and-software/dynwslib

[9] **Sockets multicast:** ejemplo implementación sockets udp con multicast. Última consulta: 30 Noviembre 2008] URL: http://www.dotnet247.com/247reference/msgs/32/164271.aspx